VALIDATION
SUMMARY
REPORT

⑫

FEDERAL
COBOL
COMPILER
TESTING
SERVICE

Department of the Navy
(ADPESO)

Washington, D.C.
20376

DDC

MAY 18 1977

B

BIBLIOGRAPHIC DATA SHEET	1. Report No. CCVS74-VSR195	2.	3. Recipient's Accession No.
4. Title and Subtitle Validation Summary Report, #CCVS74-VSR195 DECSystem-20 COBOL			5. Report Date 4 May 1977
			6.
7. Author(s) Same as organization - See 9.			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Federal COBOL Compiler Testing Service ADP Selection Office Department of the Navy Washington, D. C. 20376			10. Project/Task/Work Unit No.
			11. Contract/Grant No.
12. Sponsoring Organization Name and Address ADP Selection Office Department of the Navy Washington, D. C. 20376			13. Type of Report & Period Covered
			14.

15. Supplementary Notes

16. Abstracts

This Validation Summary Report (VSR) for the DECSYSTEM-20 (CBL74) COBOL Compiler Version 1A (TOPS-20 Version V2) provides a consolidated summary of the results obtained from the validation of the subject compiler against the 1974 COBOL Standard (X3.23-1974/FIPS PUB 21-1). The compiler was validated at the High level of FIPS PUB 21-1. The VSR is made up of several sections showing the discrepancies found. These include an overview of the validation which lists all categories of discrepancies by level/module within X3.23-1974; a section relating the categories of discrepancies to each of the Federal levels of the language; and a detailed listing of discrepancies together with the tests which were failed.

17. Key Words and Document Analysis. 17a. Descriptors

Programming Languages
Standards
Compilers
COBOL
Verifying
Proving Program Correctness
Software Engineering

17b. Identifiers/Open-Ended Terms

CCVS
CVS

17c. COSATI Field/Group 09/02		
18. Availability Statement Release unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 76
	20. Security Class (This Page) UNCLASSIFIED	22. Price

FORM NTIS-35 (REV. 3-72)   THIS FORM MAY BE REPRODUCED   USCOMM-DC 14952-P72

COBOL COMPILER
VALIDATION SUMMARY REPORT

VALIDATION NUMBER CCVS74-VSR195

Prepared By:

FEDERAL COBOL COMPILER TESTING SERVICE
DEPARTMENT OF THE NAVY
WASHINGTON, D.C.   20376

COBOL COMPILER VALIDATION
----- --------- -----------

1. Validation Number   CCVS74-VSR195

2. Vendor   Digital Equipment Corporation

3. Mainframe   DECSYSTEM-20   Model Number 2050

4. Compiler Identification   Digital Equipment Corporation
                                                    CBL74 Version 1A

5. Operating System Identification   TOPS-20 V2

6. Compiler Validation System Version Number   CCVS74 2.0

7. Federal Information Processing   21-1
     Standard Publication

*PLEASE NOTE. The Federal COBOL Compiler Testing Service may make full
and free public disclosure of the Validation Summary Report (VSR) in
accordance with the "Freedom of Information Act" (5 U.S.C. #552). The
results of this validation are only for the purpose of satisfying United
States Government requirements, and apply only to the Computer System,
Operating System release, and compiler version identified in the VSR. The
COBOL Compiler Validation System is used to determine, insofar as is
practical, the degree to which the subject compiler conforms to the Fed-
eral COBOL Standard. Thus, the VSR is necessarily discretionary and judg-
mental. The United States Government does not represent or warrant that
the statements, or any one of them, set forth in the VSR are accurate or
complete. The VSR is not meant to be used for the purpose of publicizing
the findings summarized therein.

For information concerning this compiler you can contact the vendor's
designated representative named below:

      Mr. David M. Nixon
      Digital Equipment Corporation
      200 Forest Street
      Marlboro, Massachusetts   01752

TABLE OF CONTENTS

SECTION 1.   INTRODUCTION

1.1   Purpose of the Validation Summary Report

The purpose of the Validation Summary Report (VSR) is to identify individual
COBOL language elements whose implementation does not conform to Federal Stan-
dard COBOL as adopted from American National Standard Programming Language
COBOL, X3.23-1974, by Federal Information Processing Standard 21-1 (FIPS
PUB 21-1).

1.2   Preparation of the VSR

The Validation Summary Report is prepared by analyzing the results of running
the COBOL Compiler Validation System (CCVS). The COBOL Compiler Validation
System consists of audit routines containing features of Federal Standard COBOL,
their related data, and an executive routine (VP-routine) which prepares the
audit routines for compilation. Each audit routine is a COBOL program which
includes many tests and supporting procedures indicating the result of the
tests.

The testing of a compiler in a particular hardware/operating system environment
is accomplished by compiling and executing each audit routine. The report pro-
duced by each routine tells whether the compiler passed or failed the tests in
the routine. If the compiler rejects some language elements by terminating
compilation, giving fatal diagnostic messages, or terminating execution abnor-
mally, then the test containing the code the compiler was unable to process is
deleted and the audit routine compilation and execution repeated.

The compilation listings and the output reports of the audit routines con-
stitute the raw data from which the members of the Federal COBOL Compiler
Testing Service produce a Validation Summary Report.

1.3   Organization of the VSR

The Validation Summary Report is made up of several sections the contents of
which are described below.

    a.   Section 2 summarizes the results of the compilation and execution
of the programs comprising the COBOL Compiler Validation System. Section 2
is subdivided into a subsection representing each level of each module defined
in American National Standard Programming Language COBOL, X3.23-1974. Each
subsection contains a list of all of the language elements which must be
implemented in order to claim support of that level/module. The list of
language elements will be annotated to include a description of both syntax
and semantic errors detected during the validation.

    b.   Section 3 - FIPS PUB 21-1 defines four Federal levels of the COBOL
Standard. Section 3 of the VSR lists the discrepancies described in Section 2
by the Federal level in which the problem occurs.

c. Section 4 contains information which describes the software environ-
ment in which the compiler was tested. This includes the name and version of
the operating system; the implementor-names which were used in the Environment
Division of the programs comprising the CCVS; the options used with the com-
piler; and if applicable, information regarding the use of compiler optimiza-
tion features.

d. Appendix A is the Validation Summary Working Document, a working
paper resulting from the compilation and execution of the CCVS, and from which
the VSR is derived.

1.4 Abstract Covering Compliance to American National Standard Programming
Language COBOL

Definition of an Implementation of American National Standard Programming
Language COBOL (excerpts from X3.23-1974, Chapter 1, Section 1.5).

An implementation is defined to meet the requirements of the American
National Standard COBOL specification if that implementation includes a
fully implemented specified level of each of the functional processing
modules and of the Nucleus as defined in this Standard. It follows from
this that, in order to meet the requirements of this Standard, an imple-
mentation must:

a. Not require the inclusion of substitute or additional language ele-
ments in the source program, in order to accomplish any part of the function
of any of the standard language elements.

b. Accept all standard language elements contained in a given level
of a module which is specified as being included in the implementation,
except as specifically exempted (as pertaining to specific hardware components
for which support is not claimed). See "Elements that Pertain to Specific
Hardware Components" below.

These points are of particular pertinence in two areas:

(1) There are throughout the American National Standard COBOL
specification certain language elements whose syntax, or effect, is specified
to be, in part, implementor-defined. While the implementor specifies the
constraints on that portion of each element's syntax or rules that is indi-
cated in this Standard to be implementor-defined, such constraints may not
include any requirement for the inclusion in the source program of sub-
stitute or additional language elements.

(2) When a function is provided outside the source program that
accomplishes a function specified by any particular standard COBOL element,
then the implementation must not require, except for Environment Division
elements, the specification of that external function in place of or in
addition to that standard language element:

The following qualifications apply to the American National Standard COBOL
specification:

5

a.   There are certain language elements which pertain to specific types
of hardware components. In order for an implementation to meet the require-
ments of this standard, the implementor must specify the minimum hardware
configuration required for that implementation and the hardware components
that it supports. Further, when support is thus claimed for a specific hard-
ware component, all standard language elements that pertain to that component
must be implemented if the module in which they appear is included in the
implementation. Language elements that pertain to specific hardware com-
ponents for which support is not claimed, need not be implemented. However,
the absence of such elements from an implementation of American National
Standard COBOL must be specified.

b.   An implementation of American National Standard COBOL may include
the ENTER statement or not, at the option of the implementor.

c.   An implementation that includes, in addition to a specified level of
each of the functional processing modules and of the Nucleus, elements or
functions that either are not defined in the American National Standard COBOL
specification or are defined in a given level of a standard module not other-
wise included in the implementation, meets the requirements of this Standard.
This is true even though it may imply the extension of the list of reserved
words by the implementor, and prevent proper compilation of some programs that
meet the requirements of this Standard. The implementor must specify any
optional language (language not defined in a specified level but defined else-
where in the Standard) or extensions (language elements or functions not
defined in this Standard) that are included in the implementation.

d.   In general, the American National Standard COBOL specification
specifies no upper limit on such things as the number of statements in a pro-
gram, the number of operands permitted in certain statements, etc. It is
recognized that these limits will vary from one implementation of American
National Standard COBOL to another and may prevent the proper compilation of
some programs that meet the requirements of this standard.

IMPLEMENTOR-DEFINED LANGUAGE SPECIFICATIONS

The language elements in the following lists depend on implementor defini-
tions to complete the specificaton of the syntax or rules for the elements.

The elements whose syntax is partly implementor-defined are:

Element	Implementor-Defined Aspect
SOURCE-COMPUTER paragraph	computer-name
OBJECT-COMPUTER paragraph	computer-name
MEMORY SIZE clause	integer
alphabet-name	implementor-name; whether imple-

6

                                        mentor-names are provided.

    SPECIAL-NAMES paragraph   implementor-name

    ASSIGN clause   implementor-name

    VALUE OF clause   implementor-name; whether implementor-
                                        names are provided.

    RERUN clause   implementor-name and the form; the
                                        implementor provides at least one
                                        of seven specified forms.

    CALL and CANCEL statements   relationship between operand and the
                                        referenced program.

    COPY statement   relationship between library-name
                                        text-name, and the library.

    ENTER statement   language-name

    Margin R   The location.

    Area B   The number of character positions.

    Qualification   The number of qualifiers; at least
                                        five must be supported.

The elements whose effect is partly implementor-defined are:

    Element   Implementor-Defined Aspect
    -------   --------------------------

    alphabet-name   The correspondence between native and
                                        foreign character sets.

    implementor-name switches   Whether setting can change during
                                        execution.

    USAGE IS COMPUTATIONAL   Representation and whether automatic
      clause   alignment occurs.

    USAGE IS INDEX clause   Representation and whether automatic
                                        alignment occurs.

    SYNCHRONIZED clause   Whether implicit FILLER positions are
                                        generated; their effect on the size
                                        of group items and redefining items.

    ACCEPT statement   Maximum size of one transfer of data
                                        in Level 1 Nucleus.

7

DISPLAY statement	Maximum size of one transfer of data in Level 1 Nucleus.
Numeric test	Representation of valid sign in the absence of the SIGN IS SEPARATE clause.
Comparison of nonnumeric items	Collating sequence, where NATIVE or implementor-name collating sequence is implicitly or explicitly specified.
Arithmetic expressions	Number of places carried for intermediate results.

Elements That Pertain to Specific Hardware Components

The standard language elements in the list that follows pertain to specific types of hardware components. These language elements must be implemented in an implementation of American National Standard COBOL when support is claimed, by the implementor, for the specific types of hardware components to which they pertain, and the module in which they are defined is included in that implementation.

Element	Hardware Component
CODE-SET clause	Device capable of supporting the specified code.
MULTIPLE FILE TAPE clause	Reel
CLOSE...REEL/UNIT statement	Reel or mass storage
CLOSE...NO REWIND statement	Reel or mass storage
OPEN...REVERSED statement	Reel with the capability of making records available in the reversed order; mass-storage with the capability of making records available in the reversed order.
OPEN...NO REWIND statement	Reel or mass storage
OPEN...I-O statement (Sequential I-O only)	Mass storage
OPEN EXTEND statement	Reel or mass storage
REWRITE statement (Sequential I-O only)	Mass storage

8

SEND...BEFORE/AFTER ADVANCING statement	Devices capable of vertical positioning; devices capable of action based on mnemonic-names.
USE...I-O (Sequential I-O only)	Mass storage
WRITE...BEFORE/AFTER ADVANCING	Devices capable of vertical positioning; devices capable of action based on mnemonic-name.

1.5   The Federal COBOL Standard

The COBOL compiler validation results enclosed in this document reflect the degree to which the subject COBOL compiler implements the Federal COBOL Standard.   The Federal COBOL Standard is essentially the same as the American National Standard Programming Language COBOL, X3.23-1974, with two exceptions:

    The Federal COBOL Standard defines 4 levels and the ANSI Standard defines only the minimum COBOL implementation and the full standard. Low and High levels of the Federal COBOL Standard (see 1.5.1) correspond to the above two ANSI levels (minus the Report Writer module). Two additional levels, low-intermediate and high-intermediate have been included in the Federal Standard between the highest and lowest subsets. These additional levels accommodate hardware which cannot support the full standard, but which is capable of implementing more than the minimum standard.

    The Report Writer Module is not contained in the Federal COBOL Standard.

The Federal COBOL Standard requires that a compiler contain as a minimum the elements specified in at least one of the Federal levels. No restrictions are imposed on the inclusion of selected features from higher levels or even unique vendor extensions. Compatibility among various implementations of a given level containing additional features must be controlled by management imposed standards and restrictions.

1.5.1   Federal Standard COBOL Levels

    a.   Federal Standard COBOL specifications are the language specifications contained in American National Standard Programming Language COBOL, X3.23-1974. For purposes of the Federal Standard, the modules defined in X3.23-1974 are combined into four levels. Not all computers are large enough to accommodate a COBOL compiler containing the full ANSI Standard. Therefore, the Federal Government requires that all compilers acquired by its agencies contain as a minimum one of the four Federal levels, depending on machine size, configuration and user needs. The knowledge that all computers will support at least one of these four subsets simplifies the task of developing machine-independent COBOL programs.

    b.   The four levels of Federal Standard COBOL are identified as: Low, Low-Intermediate, High-Intermediate, and High. Each Federal Standard COBOL level is composed of either the high or low levels of the nucleus and ten of the eleven Functional Processing Modules (FPMs) defined in X3.23-1974. The four Federal Standard COBOL levels are reflected in the following table. The numbers in the table refer to the level within the FPM or nucleus as designated in X3.23-1974, and a dash in the table denotes that the corresponding FPM is omitted.

10

	Low Level	Low Inter- mediate Level	High Inter- mediate Level	High Level
NUCLEUS	1	1	2	2
FPMs				
TABLE HANDLING	1	1	2	2
SEQUENTIAL I-O	1	1	2	2
RELATIVE I-O	-	1	2	2
INDEXED I-O	-	-	-	2
SORT-MERGE	-	-	1	2
REPORT WRITER	-	-	-	-
SEGMENTATION	-	1	1	2
LIBRARY	-	1	1	2
DEBUG	-	1	2	2
INTER-PROGRAM COMMUNICATION	-	1	2	2
COMMUNICATION	-	-	2	2

1.5.2 Conformance to Federal Standard COBOL

A compiler implemented in conformance to Federal Standard COBOL must meet at least the following requirements.

a. The implementation must include all of the language elements of at least one of the levels of Federal Standard COBOL.

b. The implementation must meet all of the requirements defined in American National Standard COBOL, X3.23-1974, Section I, paragraph 1.5, Definition of An Implementation of American National Standard COBOL which is provided in section 1.4 of this VSR.

c. The implementation must provide a facility for the user to optionally specify a level of Federal Standard COBOL for monitoring his source program at compile time. The monitoring will be an analysis of the syntax used in a source program against the syntax included in the specified level of Federal Standard COBOL. Any syntax used in the source program that does not conform to that allowed by the user selected level of Federal Standard COBOL will be diagnosed. The syntax diagnosed as not conforming to the specified level will be identified to the user through a diagnostic message on the source program listing. The diagnostic message will contain, at least: (1) The identification of the source program line number in which the nonconforming syntax occurs, (2) the identification of the level of Federal Standard COBOL that supports

11

the syntax or that the syntax is nonstandard COBOL.

1.6. Use of the VSR

The Federal COBOL Compiler Testing Service may make full and free public disclosure of the Validation Summary Report (VSR) in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of the validation are only for the purpose of satisfying United States Government requirements, and apply only to the computer system, operating system release, and compiler version identified in the VSR.

The COBOL Compiler Validation System is used to determine, insofar as is practical, the degree to which the subject compiler conforms to the COBOL Standard. Thus, the VSR is necessarily discretionary and judgmental. The United States Government does not represent or warrant that the statements, or any one of them, set forth in the VSR are accurate or complete. The VSR is not meant to be used for the purpose of publicizing the findings summarized therein.

1.7 Sources of Additional Information

FIPS PUB 21-1 defines the Federal COBOL Language Standard. This publication is available from the Office of ADP Standards Management, National Bureau of Standards, Washington, D. C., 20234.

The detailed COBOL language specifications are given in the publication "American National Standard Programming Language COBOL, X3.23-1974", available from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

An explanation of the COBOL Compiler Validation System is contained in the CCVS User's Guide. This document explains how to run the compiler validation system. The User's Guide and a magnetic tape containing a copy of the CCVS programs are available from the National Technical Information Service, Springfield, Virginia, 22151. (Ordering information can be obtained from the Federal COBOL Compiler Testing Service.)

1.8 Requests for Interpretation

Questions regarding this VSR or the CCVS in general should be forwarded to the FCCTS. If any problem cannot be adequately resolved through the FCCTS, the request for interpretation will be forwarded to the Federal COBOL Interpretation Committee for final resolution.

A brochure describing the validation process including the procedures for requesting a validation and resolution of questions involving interpretation of the current Federal Standard is available from the Department of the Navy, Federal COBOL Compiler Testing Service, Washington, D.C. 20376.

1.9 Federal Standard COBOL Approved Interpretation

The National Bureau of Standards published in the Federal Register Vol. 41

No. 179, September 14, 1976, an approved interpretation of Federal Standard
COBOL as pertains to the evaluation of arithmetic expressions in the COMPUTE
statements. This interpretation states that "size of the intermediate result
field is implementor-defined."

Since the results of evaluating arithmetic expressions are not predictable, all
COMPUTE statements and IF statements containing arithmetic expressions have
been removed from the COBOL Compiler Validation System.

1.10   Timeliness of the Validation Summary Reports

     The timeliness of the Validation Summary Report is important. Compilers
and their related operating system software are modified several times a year.
The Compiler Validation System used to validate compilers is also updated
during the life of the system. Therefore to ensure that the latest version
of both the vendor's compiler and the Validation System are the latest offici-
ally released versions, check with the:

          Director
          Federal COBOL Compiler Testing Service
          Department of the Navy
          Washington, D. C.   20376
          (202) 697-1247

     Please use the Validation Summary Report number of this report when
corresponding with the Testing Service.

SECTION 2.   DETAILED EVALUATION OF ERRORS.

This section summarizes the results of the compilation and execution of the
programs comprising the COBOL Compiler Validation System (CCVS). The version
of the CCVS used during this validation is shown inside the front cover of the
VSR.

Section 2 is made up of a variable number of subsections. The number of sub-
sections is dependent on the Level of Federal COBOL being validated. There will
be a subsection for each level of each module which is validated. If the high
level of a module is validated then there will be two subsections for that
module; one for the low level and one for the high level.

A validation of the low level of Federal Standard COBOL would result in three
subsections being present. One for Nucleus level 1, one for Sequential I-O
level 1, and one for Table Handling level 1.

Each error or deviation noted in this section makes reference to a program or
functional COBOL module contained in Appendix A (Validation Summary Working
Document). This reference provides the documented results of an occurrence
of errors/deviataions detected during the running of the CCVS using the compiler
within the environment identified within this document. The Validation Summary
Working Document is presented in sequence by functional module, functional
module level and program number as defined below.

Each program in the COBOL Compiler Validation System is identified by a 5-
character program name. The name associates the routine with the functional
processing module and level of American National Standard Programming Language
COBOL tested within the program.

The five character name has the general format XXNMM. The first two
characters are alphabetic and identify the functional module tested by
the program. The permissable values are:

        CM - Communication
        DB - Debug
        IC - Inter-Program Communication
        IX - Indexed I-O
        LB - Library
        NC - Nucleus
        RL - Relative I-O
        RP - Report Writer
        SG - Segmentation
        SQ - Sequential I-O
        ST - Sort-Merge
        TH - Table Handling

The third character of the audit routine name is either a 1 or 2, and
identifies the level of the functional module being tested. Each module and
level is represented by several programs. The fourth and fifth characters of
the program name are sequence numbers for programs which test features in the

14

same level of the same functional processing module.

As an example, the program name NC210 is the tenth program in the series of routines which test the second level of the Nucleus module.

Description of Section 2.

Each error/deviation is noted by number in the left hand margin opposite the language element in question. This number is used in section 3 to categorize errors by Federal level (See 1.5.1). Inserted directly below the language element is a brief description of the error. To the right of the langauge element is a page reference to X3.23-1974, American National Standard Programming Language COBOL. The reference at the end of the description of the error is to Appendix A which contains the detailed information collected during the validation. The reference is made up of the routine name followed by an A or B (A for compile time or syntax error and B for execution time or semantic error) and a number which makes the error unique in Appendix A.

Example:

2.1 Nucleus Level 1

.
.
.

2.1.9
```
      Operational symbols: S V P                          II-21
      --------------------------------------------------------
      * The scaling character 'P' is not permitted in a
      * PICTURE character-string.
      *                                        (NC101.A.2)
      --------------------------------------------------------
```

.
.
.

2.2 Sequential I-O Level 1

2.1.9 represents the ninth error for Nucleus Level 1

II-21 represents the page in X3.23-1974 where the language
element is defined

*   Boxes the description of the error/deviation

NC101.A.2 represents:

```
      Program name   -   NC101
      Syntax error   -   A
      second error   -   2
```

2.1 NUCLEUS LEVEL 1

2.1.1   --
        *   Switches are not supported.
        *   (NUCLEUS)

2.1.2   --
         *   Succeeding lines of code are rejected when the word
         *   PICTURE (or PIC) is the last word on a line.
         *   (NC157 A)

2.2   NUCLEUS LEVEL 2

All elements of 1 NUC 1,2 are a part of 2 NUC 1,2

literal

2.2.1 --
    *   The compiler did not execute correctly an INSPECT statement
    *   containing multi-character operands with TALLYING on four
    *   operands and REPLACING on six operands.
    *   (NC216 B)

2.3   TABLE HANDLING LEVEL 1

2.4 TABLE HANDLING LEVEL 2

All elements of 1 TBL 1,2 are a part of 2 TBL 1,2

2.5 SEQUENTIAL I-0 LEVEL 1

2.5.1 --
   *   The compiler does not accept the "CODE-SET IS alphabet-
   *   name" clause if the optional word 'IS' is missing.
   *   (SQ118 A.1)
2.5.2 --
   *   The compiler required a nonstandard clause (RECORDING MODE
   *   IS STANDARD-ASCII) in order to process tape files in
   *   accordance with the American National Standard Code for
   *   Information Interchange.
   *   (SQ118 A.2)

2.6   SEQUENTIAL I-O LEVEL 2

       All elements of 1 SEQ 1,2 are a part of 2 SEQ 1,2

  2.6.1   --
          *   The OPEN EXTEND statement fails to position a file to
          *   immediately following the last logical record but instead
          *   functions as though an OPEN OUTPUT statement had been
          *   specified.
          *   (SQ205 B)
          ----
             file-name series
             INPUT, OUTPUT, I-O, and EXTEND series

2.6.2   --
        *   The compiler does not accept the USE...EXTEND statement.
        *   (SQ218 A)

2.6.3   --
        *   The WRITE...BEFORE...END-OF-PAGE statement fails to
        *   recognize an end-of-page condition when the printer is
        *   positioned into the footing area of the page body.
        *   (SQ202 B)

30

2.7 RELATIVE I-O LEVEL 1

2.8   RELATIVE I-O LEVEL 2

All elements of 1 REL O,2 are a part of 2 REL O,2

2.9   INDEXED I-O LEVEL 1

2.9.1 ---
    *   A program fails to execute when restarted from a checkpoint
    *   file created through the use of the RERUN clause.
    *   (IX106 B)

2.10   INDEXED I-O LEVEL 2

All elements of 1 INX 0.2 are a part of 2 INX 0.2

2.11   SORT-MERGE LEVEL 1

2.12   SORT-MERGE LEVEL 2

All elements of 1 SRT 0,2 are a part of 2 SRT 0,2

CCVS74-VSR195

2.13   SEGMENTATION LEVEL 1

2.14 SEGMENTATION LEVEL 2

All elements of 1 SEG 0,2 are a part of 2 SEG 0,2

Environment Division
     The OBJECT-COMPUTER paragraph
          SEGMENT-LIMIT. IX-5

Procedure Division
     Segment-numbers IX-4
          Sections with the same segment-number need not
               be physically contiguous in the source program

2.15 LIBRARY LEVEL 1

2.16   LIBRARY LEVEL 2

All elements of 1 LIB 0,2 are a part of 2 LIB 0,2

2.17 DEBUG LEVEL 1

2.17.1 --
       *   The Debug Module is not supported.
       *   (DEBUG Module)

2.18   DEBUG LEVEL 2

2.18.1   --
     *   The Debug Module is not supported.
     *   (See 2.17.1)

     All elements of 1 DEB 0,2 are a part of 2 DEB 0,2

     Procedure Division
        USE FOR DEBUGGING statement. XI-4
           ALL REFERENCES OF identifier series
           file-name series
           cd-name series

2.19 INTER-PROGRAM COMMUNICATION LEVEL 1

2.20   INTER-PROGRAM COMMUNICATION LEVEL 2

All elements of 1 IPC 0,2 are a part of 2 IPC 0,2

SECTION 3.   COMPILER STATUS

Section 1.5 explains the four levels of Federal Standard COBOL. This
section lists the discrepancies described in Section 2 by the Federal level
in which the problem occurs. All errors listed for a lower level are also
errors in any higher level, even though they are listed only in the lower
level. The paragraph number from Section 2 is used to reference the errors
in each Federal level.

3.1.   Low Level

        2.1.1   Switches are not supported.
        2.1.2   PICTURE as last word on line results in error.
        2.1.4   DISPLAYed numeric data is incorrectly edited.
        2.5.1   Compiler rejects CODE-SET IS ... clause w/o 'IS'.
        2.5.2   Non-standard code required for ASCII (X3.4) tape files.

3.2.   Low-Intermediate Level

        2.17.1 Debug module is not supported.

3.3.   High-Intermediate

        2.2.1   INSPECT statement with multi-character operands and multiple
                TALLYING and REPLACING operands did not execute correctly.
        2.6.1   OPEN EXTEND failed to position to following last record in file.
        2.6.2   USE...EXTEND is not supported.
        2.6.3   WRITE...BEFORE...END-OF-PAGE... fails end-of-page condition.

3.4.   High Level

        2.9.1   Checkpoint file specified by RERUN clause unable to be restarted.
        2.10.1 ALTERNATE RECORD KEY clause is not supported.

SECTION 4   SOFTWARE ENVIRONMENT.

    The compiler referenced in this document was validated using the software
environment described in this section. When using a modification of the
described environment, the compiler may or may not continue to conform to the
Standard. It should be noted that during the validation process, an attempt
is made to validate as many different options as possible.

    The use of compiler options, implementor-names in the Environment Division
and any form of optimization which is not described in this report could cause
the compiler to produce a program that does not perform according to the
specifications of Standard COBOL. Only the environment described in this
document has been used with this compiler to satisfy the requirements of FIPS
PUB 21-1 and FPMR 101-32.1305.1a. (Any deviations which must be corrected as
per the referenced FPMR are described in Sections 2 and 3 of this report.)

1. Options or parameters used on the processor call statement for the compiler:
The following options/parameters were used during the validation.

Options specified:

        None

Options defaulted:

        N/A

2. Environment Division implementor-names.

    Printer destined files

        LPT

    Tape files

        MTAn

    Sequential Mass-storage files

        DSK

    Random Access files

        DSK

    Sort files (SD)

        DSK,DSK,DSK

48

Switch names

Not supported by the compiler.

Source Computer names

DECSYSTEM-20

Object Computer names

DECSYSTEM-20

RERUN Clause option

RERUN EVERY 10 RECORDS OF RR-FS1

3. Optimization. The compiler may or may not have optimization features. If there was an optimization feature available, it was used during the validation process (during a separate execution of the Compiler Validation System) to determine if its use causes the compiler to produce a program which does not give the expected results. If the optimization is invoked through the compiler call statement then it is mentioned in paragraph 1 above. If it is invoked through the introduction of syntax in other than the Data and Procedure Divisions of the source program it is shown below. Optimization which would require modification to the Data and Procedure Divisions is not considered in this report in that it is beyond the scope of the use of standard COBOL and the validation process.

Optimization was not considered for this validation.

4. Compiler.

CBL74 Version 1A

5. Operating system.

TOPS-20 V2

5. ASCII Validation

ASCII Validation is performed by running a sequence of three CCVS74 pro-
grams (SQ118, SQ119, SQ120) using special procedures. The purpose of this
special run is to validate that in fact the compiler/operating system being
tested is capable of processing an ASCII tape file and an ASCII card file pro-
duced (in accordance with the appropriate American National Standard) on another
system. There is also a magnetic tape and a card file created in ASCII during
the validation which will be taken to another system for further processing to
determine whether the compiler/operating system being tested can also produce
ASCII files.

This testing is based on several American National Standards and presumes
their support by the compiler/operating system being validated. These are:

1. American National Standard Programming Language COBOL X3.23-1974

   - The CODE-SET clause is used to read and write the ASCII
     files.

   - The PROGRAM COLLATING SEQUENCE clause is used to process the
     data in ASCII mode as well as native mode.

   - The SIGN...SEPARATE clause is used for signed data and all
     data is in the DISPLAY (character) mode.

2. American National Standard Code for Information Interchange
   (ASCII) X3.4-1968. (Note that this describes the code, not the
   labeling and tape recording formats.)

3. American National Standard Hollerith Punched Card Code,
   X3.26-1970.

4. American National Standard Magnetic Tape Labels for Information
   Interchange, X3.27-1969.

5. American National Standard Recorded Magnetic Tape for Information
   Interchange (800 CPI, NRZI), X3.23-1967.

6. American National Standard Recorded Magnetic Tape for Information
   Interchange (1600 CPI, PE), X3.39-1973.

The language of the 1974 COBOL Standard provides the capability to accept,
process, and produce ASCII code. The ASCII Standard describes the code insofar
as the bit arrangement and configuration, but does not address recording tech-
niques, record formats or any labeling scheme. The 800 CPI, NRZI magnetic tape
recording standard was used to establish the recording density and techniques.
(1600 CPI, PE based on X3.39-1973 "Recorded Magnetic Tape for Information Inter-
change" could be used under special arrangements.) The tape labeling scheme
used in these tests is based on X3.27-1969 but is also compatible with the
revision to that tape label standard. Only the VOL1, HDR1, and EOF1 labels are
used. The records are fixed length and unblocked.

50

During the validation, the Validation Manager for the Federal COBOL Compiler Testing Service would use the necessary ASCII magnetic tape and card files in addition to the normal tape files associated with a validation. For the ASCII portion of the validation the following steps are performed:

1. The tape file and card deck (produced on another computer system) are used as input to several programs designed to validate whether the system being validated can accept and process ASCII data. Any changes made during this validation to the source programs reading the data are noted below.

2. A tape file and card file are produced during the validation which should prove to be identical to the files described in 1 above. These two files are then processed on a different computer system to determine the degree to which the system being validated supports the ASCII standard. Any changes made during this validation to the source program producing the data are noted below.

Results for this Validation

The system was able to produce but unable to read the required ASCII tape files based on reference 2 above. Only the first record from the input magnetic tape file provided by FCCTS was processed before an end-of-file condition was encountered. A total of 50 records should have been processed.

On the other hand, the output tape file was created with an additional character preceeding each record on the file. This resulted in each record being offset by one character when the file was verified.

The 800 CPI NRZI magnetic tape recording format was used for this validation (reference 5 above). The following modifications had to be made in order to fulfill the ASCII validation requirement.

1. The tape labeling standard (reference 4 above) is not supported by the Operating System used during the validation. An unlabeled tape file was used as input and the LABEL RECORDS clause in each of the source programs was modified to reflect OMITTED labels.

2. The nonstandard clause "RECORDING MODE IS STANDARD-ASCII" was added to the SELECT statement of each of the tape files in order to use the ASCII code described in reference 2 above. (Note - The CODE-SET clause in the File Description Entry which should have accomplished this function did not, thus the introduction of the nonstandard RECORDING MODE clause.)

51

APPENDIX A

VALIDATION SUMMARY WORKING DOCUMENT

A-1   This appendix is a working paper produced during the validation
and documents the results of the compilation and execution of each of the
programs comprising the CCVS. The results contained herein are based on
the use of the compiler within the Validation Environment identified in
this appendix. This appendix (Validation Summary Working Document) is not
part of the official Validation Summary Report (VSR) and is not intended
to reflect in any way the compiler's usefulness or degree of conformance
to the language specifications.

        The reader of this appendix should keep in mind that the same pro-
blem area may appear in more than one program, but is considered only as one
single discrepancy and as such is reflected only once in the body of the
VSR. (The VSR will in turn only reference the first occurrence of the
problem in the appendix.)

        This appendix is divided into two parts. The first part describes
the Validation Environment. The second part of the document is divided into
categories of information: compilation and execution results.

        The reference document for COBOL is FIPS PUB 21-1 (X3.23-1974).

52

VALIDATION ENVIRONMENT

COMPILER IDENTIFICATION: DIGITAL EQUIPMENT CORPORATION, CBL74 VERSION 1A

COMPUTER SYSTEM: DECSYSTEM-20   MODEL 2050

OPERATING SYSTEM: TOPS-20 V2

DEBUG MODULE

DB101 thru DB105, DB201 thru DB204

     The test set for this module was not run since the standard DEBUG MODULE
has not been implemented in this compiler.

INTER-PROGRAM COMMUNICATION, LEVEL 1

IC101 thru IC115, IC151, IC152

    A. Compilation

        No errors.

    B. Execution

        No failures.

INTER-PROGRAM COMMUNICATION, LEVEL 2

IC201 thru IC208

    A. Compilation

        No errors.

    B. Execution

        No failures.

INDEXED I-O, LEVEL 1

IX101 thru IX105

    A. Compilation

        No errors.

    B. Execution

        No failures.

IX106

    A. Compilation

        No errors.

    B. Execution

        The program fails to execute when restarted from the checkpointed file
        created through the use of the RERUN clause.

IX107

    A. Compilation

        No errors.

    B. Execution

        No failures.

INDEXED I-O, LEVEL 2

IX201 thru IX204

    A. Compilation

        No errors.

    B. Execution

        No failures.

IX205 thru IX208

    A. Compilation

        In IX205 through IX208, all references to ALTERNATE RECORD
        KEYs were rejected as "IMPROPER CLAUSEs".

    B. Execution

        IX205 through IX208 were not run because of ALTERNATE KEY tests
        (not supported).

LIBRARY, LEVEL 1

LB101 thru LB107

 A. Compilation

  No errors.

 B. Execution

  No failures.

LIBRARY, LEVEL 2

L3201 thru LB205

    A. Compilation

       No errors.

    B. Execution

       No failures.

LB206

    A. Compilation

       No errors.

    B. Execution

       LB206 had one test failure because that test involved the use of debug lines which are not supported by the compiler.

L3207

    A. Compilation

       No errors.

    B. Execution

       No failures.

NUCLEUS, LEVEL 1

NC101 thru NC108

    A. Compilation

    All references to hardware switches were optioned out prior to
    validation since the system does not provide switches.

    B. Execution

    No failures.

NC109

    A. Compilation

    No errors.

    B. Execution

    In NC109, NC157 and NC204 all tests involving the DISPLAYing of numeric
    data items failed. The correct and computed results are as follows:

TEST	CORRECT RESULTS	COMPUTED RESULTS
DISP-TEST-3	0123456789	123,456,789
DISP-TEST-7	...1C0123456789A...	...1C123,456,789A...
DISP-TEST-13	... E 0102030405 ...	... E 12345 ...

NC110 thru NC120, NC151 thru NC156

    A. Compilation

    All references to hardware switches were optioned out prior to
    validation since the system does not provide switches.

    B. Execution

    No failures.

NC157

    A. Compilation

    In NC157 and NC205 when the word PICTURE (or PIC) is the last word
    on a line of source code within the PICTURE clause the compiler rejects
    the succeeding line of source code indicating that a PICTURE and
    positive integer are required. In NC157 the PICTURE clause and LONG-

TEST-3 which tests this construct were deleted in order to obtain a clean compile. In NC205 all blank lines within the flagged PICTURE clauses were deleted.

   B. Execution

   No failures.

NC158

   A. Compilation

   No errors.

   B. Execution

   See NC109.

NC159 thru NC165

   A. Compilation

   No errors.

   B. Execution

   No failures.

NUCLEUS, LEVEL 2

NC201 thru NC203

    A. Compilation

        No errors.

    B. Execution

        No failures.

NC204

    A. Compilation

        No errors.

    B. Execution

        See NC109.

NC205

    A. Compilation

        See NC157.

    B. Execution

        No failures.

NC206 thru NC215

    A. Compilation

        No errors.

    B. Execution

        No failures.

NC216

    A. Compilation

        No errors.

    B. Execution

        In NC216, the following INSPECT test is executed:

```
INSPECT identifier-1 TALLYING identifier-2 FOR ALL 'A'
      identifier-3 FOR LEADING 'AH'
      identifier-4 FOR CHARACTERS BEFORE '.'
      identifier-5 FOR CHARACTERS AFTER 'AL'
      REPLACING
           FIRST 'L ' BY 'ZZ' AFTER INITIAL 'AL'
           FIRST 'BAD' BY 'ZZZ' AFTER 'L '
           LEADING 'BAD' BY 'ZZZ' BEFORE INITIAL 'Q'
           FIRST 'BAD' BY 'ZZZ' BEFORE INITIAL 'Z'
           FIRST 'BAD' BY 'ZZZ' AFTER 'ALL '
           ALL '.' BY 'Z' AFTER 'AL'.
```

The initial contents of identifer-1, PICTURED as X(83), is:

AH YES AH YES W.C. FRITOES HERE. ANYONE WHO HATES
DOGS AND KIDS CAN NOT BE ALL BAD.

The test gave the following error indications:

	COMPUTED VALUE	CORRECT VALUE
identifier-1 (last eight char. final value)	ALZZBADZ	ALZZZZZZ
identifier-3	1	0
identifier-4	15	13
identifier-5	6	5

For identifer-1, since the boundary for comparison in the second
REPLACING phrase (AFTER 'L ') is established before 'L ' is
replaced by 'ZZ' (first REPLACING clause), 'BAD' will indeed
be matched and replaced by 'ZZZ'. [See II-71, 6(c).]

For identifier-3, the second TALLYING for leading 'AH' will
never participate in a comparison because each 'A' in the first
TALLYING clause operation is matched prior to the 'AH' comparison.
The match of 'A' terminates the comparison cycle on the 'A' and
starts a new comparison cycle on the character that is immediately
to the right of the matched 'A'. [See II-70 5(c) and II-71 6(c).]

For identifier-4, whenever an 'A' match is made by the first
TALLYING clause, the third TALLYING clause is not performed, so
the count of "CHARACTERS BEFORE '.'" is 13 (or 15 - 2) and not
15.

For identifer-5, and for the same reason as above, the number
of "CHARACTERS AFTER 'AL'" is 5 and not 6, since the 'A' in
the last six characters of the string is matched by the first
TALLYING clause, and thusly is not eligible for matching by the
fourth TALLYING clause.

NC217, NC218

64

A. Compilation

    No errors.

B. Execution

    No failures.

RELATIVE I-O, LEVEL 1

RL101 thru RL109, RL151 thru RL153

    A. Compilation

       No errors.

    B. Execution

       No failures.

RELATIVE I-O, LEVEL 2

RL201 thru RL205

    A. Compilation

        No errors.

    B. Execution

        No failures.

SEGMENTATION, LEVEL 1

SG101 thru SG106

    A. Compilation

       No errors.

    B. Execution

       No failures.

SEGMENTATION, LEVEL 2

SG201 thru SG204

    A. Compilation

        No errors.

    B. Execution

        No failures.

SEQUENTIAL ACCESS, LEVEL 1

SQ101 thru SQ117

    A. Compilation

        No errors.

    B. Execution

        No failures.

SQ118, SQ119, and SQ120

    A. Compilation

        1. In SQ119 and SQ120, fatal errors were flagged on the construct

            "CODE-SET ASCII-CODE"

        located in the file description (FD) for a sequential file. In
        each case when the optional word 'IS' was added the compiler
        accepted the syntax of the statement containing the CODE-SET
        clause.

        2. In SQ118, SQ119, and SQ120 the nonstandard clause "RECORDING
        MODE IS STANDARD-ASCII" had to be added to the SELECT statements
        for the files which produced/read ASCII tape files during the
        ASCII validation of the Compiler/Operating System. The CODE-
        SET clause in the File Description Entry was present for these
        tape files and the use of the nonstandard clause should not
        have been necessary.

    B. Execution

        No failures.

SQ121 and SQ151 thru SQ153

    A. Compilation

        No errors.

    B. Execution

        No failures.

SEQUENTIAL ACCESS, LEVEL 2

SQ201

    A.   Compilation

        No errors.

    B.   Execution

        No failures.

SQ202

    A.   Compilation

        No errors.

    B.   Execution

        In SQ202, SQ213, and SQ214 the execution of all WRITE...BEFORE...
        END-OF-PAGE statements failed to execute the end-of-page phrase on
        an end-of-page condition when the printer was positioned into the
        footing area of the page body.

SQ203 and SQ204

    A.   Compilation

        No errors.

    B.   Execution

        No failures.

SQ205

    A.   Compilation

        No errors.

    B.   Execution

        SQ205 creates magnetic tape and mass storage files of 750 records
        each. Then each file is closed and reopened with an OPEN EXTEND
        statement and an additional 250 records are written on each file.
        SQ205 failed to execute properly when a premature end-of-file exit
        was taken on a normal read operation.

SQ206 thru SQ212

    A.   Compilation

71

No errors.

B. Execution

No failures.

SQ213 and SQ214

A. Compilation

No errors.

B. Execution

See SQ202.

SQ215 thru SQ217

A. Compilation

No errors.

B. Execution

No failures.

SQ218

A. Compilation

The compiler rejected as invalid the statement:

USE AFTER ERROR PROCEDURE EXTEND

indicating that it was in conflict with a second USE statement which followed it. This USE statement and SEQ-TEST-001 which tests this construct were deleted in order to obtain a clean compile.

B. Execution

No failures.

CCVS74-VSR195

SORT-MERGE MODULE, LEVEL 1

ST101 thru ST117

    A. Compilation

        No errors.

    B. Execution

        No failures.

73

CCVS74-VSR195

SORT-MERGE MODULE, LEVEL 2
ST201 thru ST215

    A. Compilation

       No errors.

    B. Execution

       No failures.

CCVS74-VSR195

TABLE-HANDLING MODULE, LEVEL 1
TH101 thru TH111, TH151, TH152
  A. Compilation
     No errors.
  B. Execution
     No failures.

CCVS74-VSR195

TABLE-HANDLING MODULE, LEVEL 2

TH201 thru TH220

 A. Compilation

   No errors.

 B. Execution

   No failures.